Requirements and Analysis Document

Viktor Franzen, Tobias Lindroth, Spondon Siddiqui, Alexander Solberg

October 11, 2018

1 Introduction

The purpose of the project is to design and create a messaging application in which multiple users can communicate with another. To differentiate from other messaging applications, the main purpose is that multiple users can join a group to communicate primarily on desktops or laptops. Students working in groups will benefit from the application as it provides a simple, private space for them to communicate and share information. As group work is common within education, the application serves a purpose for higher education and is therefore required to simplify communication between groups.

The application will be a cross-platform desktop application with a graphical user interface. The aim is develop the application so it is easy and pleasant to use, as well as have the possibility of being extended into a more complex messaging application. First time users get the possibility to enter the application, identify themselves, and thereafter search for groups they wish to join and communicate with. Once active, the user can view, send, and receive messages.

1.1 Definitions, acronyms, and abbreviations

GUI - Graphical User Interface Channel - A chat group

2 Requirements

2.1 User Stories

All the user stories will have the non-functional requirements:

Documentation - Is the code well documented? Testability - Can the code be tested in some way?

Story Identifier: STR00 Story Name: Send message

Description

As a user I want to be able to send a message to a channel so that other users in the channel can see it.

Acceptance criteria

Functional

- A user can send a message to a channel
- A user can't send an empty message to a channel
- There is a field where the user can write messages
- There is a send button
- All group members receive messages from the channel

(-A user can join a channel)

Non-functional

- Security - Are users that are not members of a channel prevented from seeing the channel's messages?

Story Identifier: STR03 Story Name: Create public group

Description

As a user I want to be able to create a public group conversation so that multiple people can talk together

Acceptance criteria

Functional

- There is a button the user can press to create a new group
- There is a field where the user can give the group a name
- There is a field where the user can give the group a description
- Other users can join the group

Non-functional

- Response time - Can other users join the channel directly after it is created?

Story Identifier: STR09 Story Name: Overview of channels

Description

As a user I constantly want to to see an overview of all my conversations so i can keep track of the channels I'm active in.

Acceptance criteria

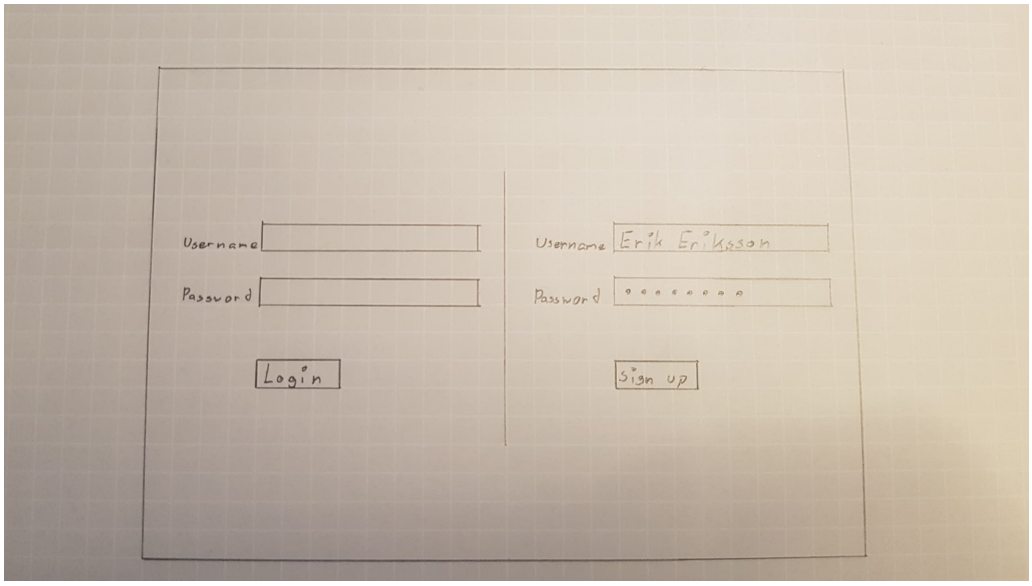
Functional

- There is a list with the channels the user is a member in
- The list is updated when joining a channel
- The list is update when leaving a channel

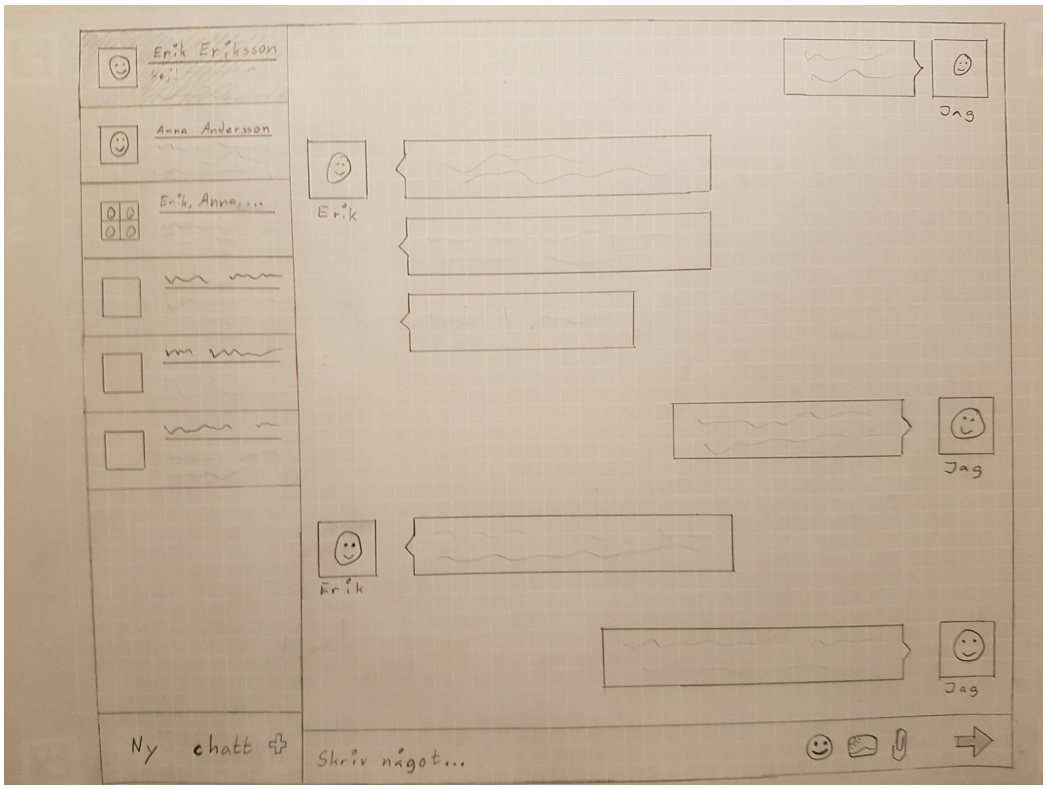
Non-functional

- ...

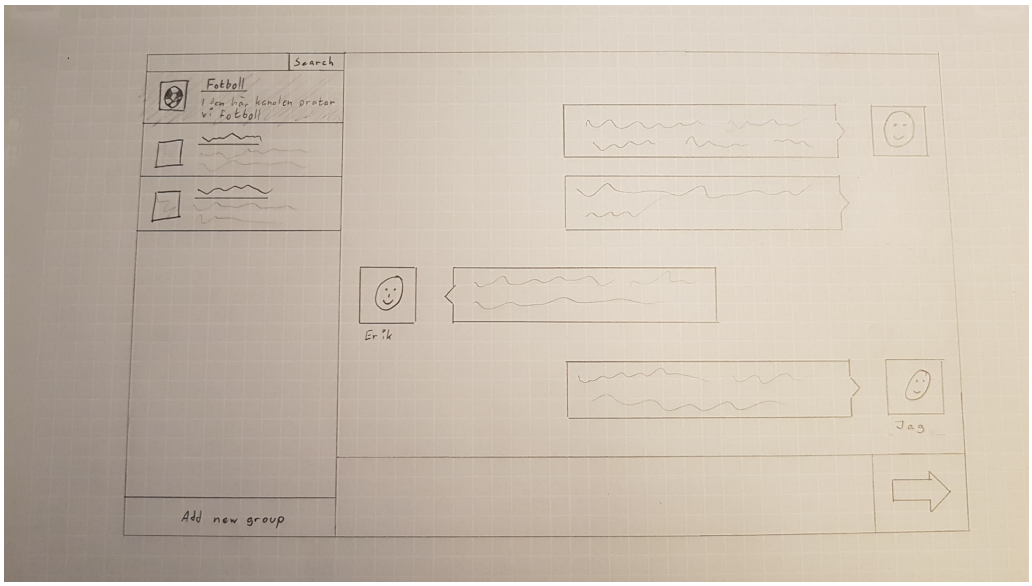
2.2 User interface



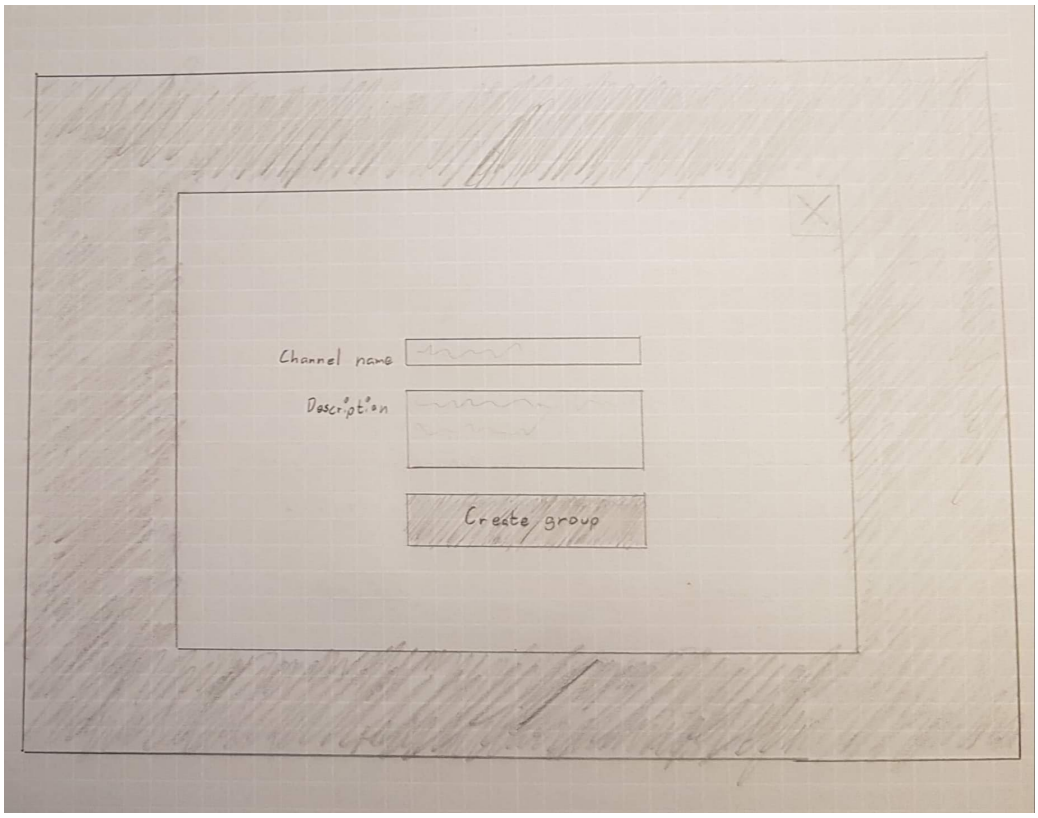
When the application is started the user is met with a view for logging in or signing up. If the user has an existing account, they can can use the fields to the left to log in to it. If they wish to create a new account, they can do so using the fields to the right. The user has to type in a username and a password for either of these to work.



This is the first iteration of the main view, which will show after the user has logged in or signed up. Here, they can view the channels they are in to the left, and the channel they are currently in is shown to the right/center of the screen. Here, the history of the channel's messages are shown, and the messages are displayed next to the profile picture of the user who sent them. The field at the bottom of the screen is used for typing in messages, and the arrow-button in the bottom-right corner is used for sending messages. The button in the bottom-left corner is used for creating a new channel.



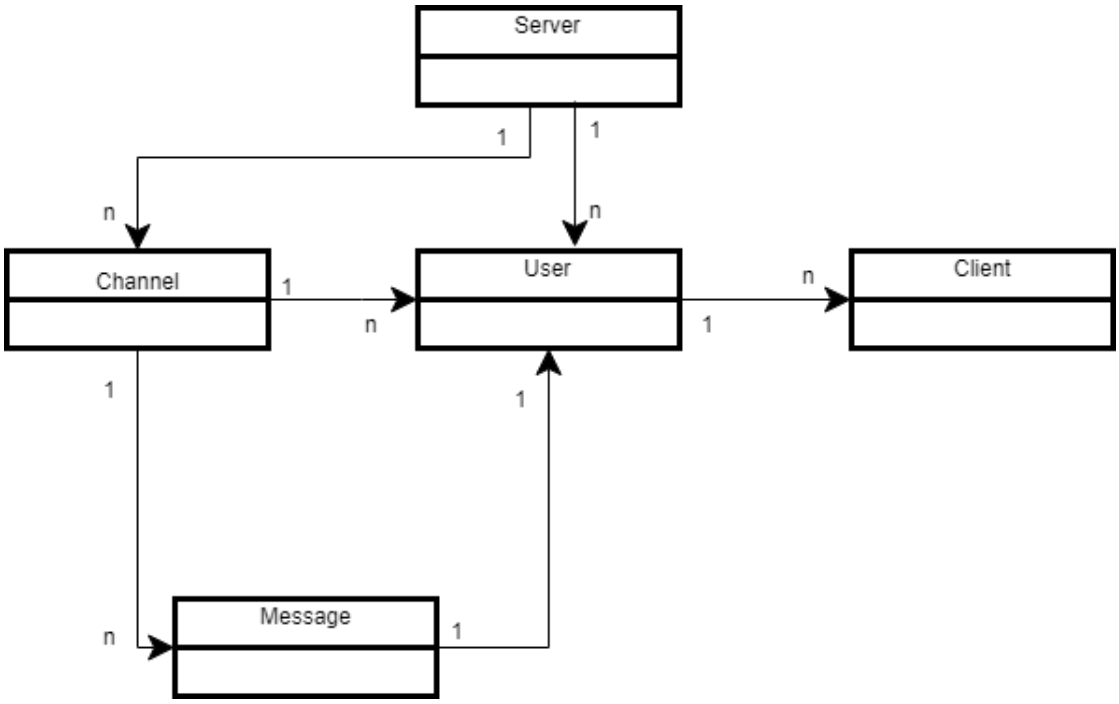
This is the second iteration of the main view. The application has been made wider and less square-ish and the channels are now displayed differently. Now, the name and the description of the channel is shown, as opposed to the users and the latest messages of the channel. Furthermore, a searchbar has been added to the top-left corner to allow users to search for a desired channel.



This is the view for creating new channels. The view is shown when the user presses the "Add new group" button at the bottom-left of the main view, and is shown on top of the main view. Moreover the main view is darkened to highlight the creation-view. In order to create a new channel, the user has to type in a name and a description for the channel. The user can choose to cancel the creation by either clicking on the x-button in the top-right corner, or by clicking outside on the view, on the main view behind it. The user is then brought back to the main view.

3 Domain model

The domain model consists of five objects. Server, channel, user, client and message. Together, they form the core of the application.



3.1 Class responsibilities

The server class is responsible for keeping track of all the existing users and channels.

The channel class holds all the messages written to it and keeps track of all the members. You can use the channel to send a message, and then the channel is responsible for delivering this message to all the members of the channel.

The user class is responsible of keeping track of its clients and forwarding all the received messages to the clients.

The client is responsible for being observable and forwarding any messages it receives to any eventual listener.

The message class is responsible for all the information regarding a message. The message content, who sent it, and when the message was created.

4 References